Minecraft Settlement Generator Description

Eduardo Hauck[1], Claus Aranha
University of Tsukuba, Department of Computer Sciences, Japan

UrbanSettlementGenerator[2] aims at generating modern cities given a Minecraft map. It generates houses and apartment buildings, connecting all of them with roads. The buildings are generated according to a downtown/outskirts division, with apartment buildings placed towards the center and houses placed towards the edges of the map.

Houses have walls made of double stone slab and pitched roofs made of wood. They have one oak door as the entrance on one random side, and glass windows on the walls perpendicular to the door wall. Houses are furnished with carpets, a table in the center of the room, a chandelier on the ceiling, and a bed, bookshelf and couch in the corners.

Apartment buildings have clay walls, 4 to 10 floors, with one apartment on each floor. The entrance to each building leads to a small corridor with a door to the ground floor apartment and a stairwell for reaching other floors. The apartments' furniture follows a similar fashion as in the houses, and windows are on the opposite wall to the building entrance.

The generator takes a number of steps in order to generate the city. First, it takes the space received in the perform function and generates a height map, a 2D matrix containing the height of the first valid ground block for all coordinates x, z of the map. It then gets the width and depth of the space, and selects a percentage of the central area to become the city center. The remaining areas are divided into 4 sections to become neighborhoods.

The center and the 4 neighborhoods go through binary or quadtree (chosen randomly each time) space partitioning 100 times and the generated partitions are added to a list. Each partition becomes a building lot, and the ones that have water, lava, etc in their perimeter or do not reach the set minimum size are eliminated as invalid. The list is then ordered according to the steepness of the lots. The steepness is calculated as the sum of differences between the most occurred height (according to the height map), and the height of every other block in the lot. Therefore, a steepness of 0 means a building lot with a flat surface.

The construction starts by pulling a building lot from the list, checking if it does not intersect with any previously pulled lot, and building a house or apartment building on it. This procedure is repeated until reaching a certain minimum number of lots for each of the 5 sections.

[1] eduardohauck@gmail.com
[2] https://github.com/ehauckdo/UrbanSettlementGenerator

To generate a building, the first thing to be done is to check whether the lot terrain is flat. If not, earthworks are carried out to ensure that the perimeter is at the same height level. This is done by finding the most occurred height, and then flattening the entire lot to match that. Grass blocks are used as the base block for flattening.

Once a lot is flattened, generation proceeds at that height. The generation of buildings and houses are mostly hardcoded. For the houses, it is as follows: first, the land is cleared changing trees etc to air blocks. Then a random width and depth between a certain range, and the maximum height of the house are set. From that, floor and walls are generated, an orientation (N/S/E/W) is decided based on the position of the house, always facing the center of the map. The orientation will dictate the location of the entrance door, the entrance to the lot at the edge of it where roads will "connect", as well as the position of the windows. The orientation also sets the direction of the pitched roofs. Once the house is completed, the interior is furnished. For apartment buildings, generation happens in a similar way.

Once all the buildings have been placed, the last step is connecting them with roads. To perform this, we consider each lot as a node in a graph, and we create a minimum spanning tree to connect them. Distances between nodes are calculated with Manhattan distance. Once we have the minimum spanning tree, we use A* to actually find the path to generate the road. The heuristic function that gives the cost between the current and the target point is the Manhattan distance. The cost function between each node is $1+n^2$, where n is the difference of height between the current block and the next. We employed this function to try and find a path that is not too steep between two points (e.g. going around a mountain instead of climbing it).

The A* returns a sequence of coordinates 1 block wide where the pavement is to be generated. We complete the road on both sides of this path to make it 3 blocks wide. When doing that, we verify if the neighbouring block is not an invalid block (e.g. water), and if it is on top of air, we fill the blocks below recursively until getting to a valid ground block.

Some of the things the we attempted include employing A* for generating the MSP, which proved to be computationally expensive for a big number of nodes. We also tried using the most occurred block of a building lot as the base block of the lot when flattening (as opposed to defaulting to grass blocks), which created some unexpected and/or visually unappealing results, like houses built on top of sand.